LBL-30613

Presented at the U.S. DECUS Spring 1991 Symposium,
Atlanta, GA, May 6–10, 1991, and to
be published in the Proceedings

A Complex Multitasked Data Acquisition and Control System for Measuring Window Thermal Efficiency, or How TSX+ Saved Our Project When It Outgrew RT-11

M. Yazdanian, J.R. Michelson, and G.O. Kelley

May 1991

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

LBL-30613
Mo-282

Presented at the U.S. DECUS Spring 1991 Symposium, May 6-10, 1991, Atlanta, GA, and published in the Proceedings.

A Complex Multitasked Data Acquisition and Control System For Measuring Window Thermal Efficiency Or How TSX+ Saved Our Project When it Outgrew RT-11

Mehrangiz Yazdanian, J. Randy Michelson, and Guy O. Kelley
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

May 1991

This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Building Technologies, Building Systems and Materials Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

A Complex Multitasked Data Acquisition and Control System For Measuring Window Thermal Efficiency Or How TSX+ Saved our Project When it Outgrew RT-11

Mehrangiz Yazdanian, J. Randy Michelson, and Guy O. Kelley
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

ABSTRACT

The Mobile Window Thermal Test facility (MoWiTT) predicts the energy efficiency of windows by continuously measuring the net heat transfer into and out of two room-sized calorimeters. The system consists of a data acquisition/control unit which communicates through a General Purpose Interface bus (GPIB IEEE-488) with an LSI-11/23 computer. We wrote the software in FORTRAN 77 and run it under TSX+ using detached jobs, the message facility, and shared run time systems to execute nine 64KB jobs which simultaneously take data and store it on disk. We arbitrate the access to a single scanner on the GPIB by running a SERVER task which acts as a virtual machine to serialize the requests. We can control and monitor the system remotely and run a number of interactive jobs without interfering with the data collection tasks. We cover a description of the hardware; the software requirements, including the libraries to control the data collection and drive the hardware; the system architecture, including a block diagram; design considerations to meet real-world requirements; and the problems we discovered along the way and the solutions we found either to correct or to bypass them.

INTRODUCTION

To measure the energy efficiency of different windows under realistic conditions, we built a Mobile Window Thermal Test facility (MoWiTT)[1)] at the Lawrence Berkeley Laboratory (LBL). It consists of dual, guarded, room-sized calorimeters in a mobile structure. The MoWiTT is capable of simultaneously exposing two window samples, each seeing a roomlike interior environment, to ambient outdoor weather conditions, and measuring the net heat transfer through each window with good accuracy.

Each calorimeter chamber contains an electric heater, a liquid-to-air heat exchanger, and a nearly continuous interior skin of large area heat flow sensors.

After the initial testing and calibration, we moved the MoWiTT to the University of Nevada campus in Reno, where the extreme seasonal weather conditions provide an ideal test site.

[1)] Klems, Joseph H.; Selkowitz, Stephen; and Horowitz, Sy. 1982. "A mobile facility for measuring net energy performance of windows and skylights." An Foras Forbartha, Energy Conservation in the Built Environment (Proceedings of CIB W67 Third International Symposium), Vol. III, p.3.1, Dublin, Ireland.

The MoWiTT system has a data acquisition/control unit[2] which communicates through a General Purpose Interface Bus (GPIB IEEE-488)[3] with an LSI-11/23 which collects and processes the data.

The computer collects data at short intervals, 24 hours a day, from temperature sensors, anemometers, radiometers, and many other instruments (as many as 270 signals). It then averages the data over ten minute intervals and stores it on disk for further analysis. We then automatically transfer the data back to computers at LBL on a daily basis via modem[4]. We also control and monitor the system remotely from LBL, 218 miles away.

When we originally designed the MoWiTT system, we chose the LSI-11/RT-11 system because, at the time, it was less expensive than a VAX and easier to use than RSX, while still providing genuine real-time capabilities. As the code grew bigger, we faced a major problem: the code did not fit within the 16-bit address space limitation of RT-11.

We solved this problem by installing TSX+[5], a high performance operating system for DEC's PDP-11 and LSI-11 computers. It provides a multiuser/multitasking programming environment similar to the RT-11 XM (extended memory) monitor. The biggest advantages of using TSX+ are that most of the code written for RT-11 is directly usable under TSX+, and that the TSX+ working environment is almost identical to RT-11. Both made the conversion a lot easier.

At the present time, we run nine 64KB programs simultaneously as detached jobs which exchange information through a shared common block, and synchronize using the TSX+ message communication facility. In addition to these detached jobs, we usually run a few interactive tasks. The system is still capable of handling additional jobs. To utilize the multitasking features of the TSX+ system, we use a number of subroutines from TSXLIB[6]

In the process of writing the software, we discovered and overcame a number of problems with the TSX+ operating system, the GPIB software library, and the ADLIB[7] software, which runs the four channel 16-bit binary input pulse counters[8]

We had to carefully select, fine tune, and extensively test the TSX+ system parameters to ensure correct handshaking between the separate detached jobs.

HARDWARE DESCRIPTION

The computer consists of the following pieces:

1 – An LSI-11/23 with 4MB of memory.
2 – Two RX02-equivalent floppy drives from Qualogy.
3 – Two RL02 drives.
4 – A GPIB11V-1 interface card from National Instruments.

2) Hewlett-Packard 3497A Data Acquisition/Control Unit.

3) GPIB11V-1 interface card from National Instruments.

4) Will be discussed in a future paper.

5) TSX+ is a registered trademark of S&H Computer Systems Inc., Nashville, TN.

6) A library of FORTRAN-callable routines that implement programmed requests for TSX+; N.A. Bourgeois Jr., SystemEngineering Division, Sandia National Laboratories, Albuqurque, NM.

7) Analog and Digital I/O Fortran Subroutine Library from ADAC Corporation; Woburn, MA.

8) Model 1604/OPI from ADAC Corporation.

5 – Three 1604/OPI four channel 16-bit binary input pulse counter cards from ADAC Corporation which measure the flow rates of water in cooling loops and the wattage of heaters and fans which control chamber temperatures.
6 – Two 1412 four-channel D/A interface cards from ADAC Corporation to drive a chart recorder.
7 – One DLV11-J 4-line serial interface card.
8 – One DZV11 4-line multiplexed serial interface card.
9 – One Supra 2400 baud modem, and some terminals and printers.

The HP 3497A scanner (a multiplexing digital voltmeter with GPIB interface) is the main data acquisition and control unit. The LSI-11 controls and reads the scanner through the GPIB11V-1 card. The scanner reads 180 thermistors (which measure temperatures at various points), 26 chamber-wall heat flow sensors, several weather sensors (such as wind speed and direction) and various solar energies.

SOFTWARE DESCRIPTION

We wrote the MoWiTT software in FORTRAN 77 (except for a few small macro routines) and currently run it under TSX+ V6.2 (using RT-11 V5.1c utilities). MoWiTT also uses a number of software libraries, including:

1 – TSXLIB V5.1 library routines for TSX+ (DECUS #11-490), as modified by Warren Massey of LLNL for full compatibility with F77.
2 – GPIB utility routines, revision D.10, from National Instruments to drive the GPIB11V-1 interface board.
3 – ADLIB (Jul-15-80 version) software package to drive the ADAC pulse counter and D/A boards.
4 – TTLIB (Aug-12-81 version; DECUS #11-495) library routines which provide the ANSI escape sequences for controlling the VT-100 terminals.

SYSTEM ARCHITECTURE

In order to provide direct, continuous measurements of the net energy flow into or out of the test chambers, MoWiTT makes related but independent readings of:

1 – various air and surface temperatures throughout the test chambers (MSTSCN job);
2 – heat flow through the surfaces surrounding the test chambers (HFM job);
3 – solar intensities and wind data (WETHER job);
4 – power to the heaters and fans which stabilize the interior air temperature in the test chambers (WATMTR job) ; and
5 – liquid-to-air heat-exchanger flow rate and inlet/outlet temperatures (FLOMTR job) [water-flow calorimetry].

Most of the jobs take readings every five seconds, averaging them over ten minute intervals to compress the data to a manageable size. All the jobs except WATMTR need to access the HP Scanner on the GPIB to collect the required data. WATMTR and FLOMTR also need to read the ADAC pulse counters.

We originally tried to run under RT-11 FB. We wrote a main scheduling program with each of the above tasks as individual subroutines. The main program controlled scheduling using RT-11 timer completion routines. The code grew bigger as we added more measurements, and soon exceeded the maximum address space available under RT-11 FB. The arbitration of access to the

HP Scanner on the GPIB also got very complex, because some of the algorithms required indivisible sequences of multiple GPIB operations.

We found that TSX+ could help us solve our biggest problems with the least amount of reprogramming effort. We relieved the address space limitations of the single large program by splitting the program into a number of interrelated detached jobs. We simplified the HP Scanner arbitration code significantly using the TSX+ message facility's inherent queuing mechanism.

All the data acquisition tasks now run as separate detached jobs. To enable these jobs to share the single HP Scanner on the GPIB, we wrote the SERVER process that arbitrates GPIB access for these conflicting tasks by emulating a GPIB virtual machine. The critical concept of the SERVER process is that it executes virtual instructions that represent one or more indivisible sequences of GPIB commands to carry out collection of data. This allows other jobs to continue to run without having to lock the CPU for the duration of each virtual instruction, some of which can require several seconds to complete.

We use the TSX+ message facility to send requests for execution of GPIB virtual instructions to the SERVER job through a single message channel which serializes the requests. The SERVER job executes each request indivisibly, and then passes the measurements back to the requesting jobs through separate message channels.

There are three other detached jobs running: the DSKWRT job, which writes the data to the disk; the DSKCHK job, which monitors the free space in the current data file and opens a new one when the current one fills; and the ERRMSG job, which checks for any error messages flagged by the data collection jobs and writes them to the disk.

The detached jobs have no direct impact on one another, no timing conflicts, and are slow enough that the CPU can process them all and still have significant (40%) idle time on our LSI-11/23.

The data acquisition jobs process the data they receive from the SERVER job, and pass the results on to DSKWRT, which stores it on disk. They pass the data through a shared common block which is simultaneously mapped by most of the various detached jobs.

We start, stop, and monitor the detached jobs through a command interpreter kernel which is an IND control file (CNTROL.COM). The interpreter can also activate one of a number of interactive jobs which perform such tasks as changing the MoWiTT initial constants (like the sampling periods of the data acquisition), displaying selected sections of the data, writing comments in a log file, and controlling the storage of instantaneous measurements (raw data for debugging) on disk. Some of these jobs access the shared common block to display or process the data and others read the data files to get the information they need. (We use the TSX+ shared-file system services to avoid any file conflicts.)

There are other operator-interface jobs that the user can activate through the command interpreter kernel to monitor, control, and view the data while data acquisition is in progress. Multiple copies of the command interpreter can run various data display and analysis jobs simultaneously. In fact, we often display the data simultaneously in both Reno and Berkeley using two copies of the same task.

DESIGN OF MoWiTT SOFTWARE

We run our data acquisition programs as detached jobs because they do not have any terminal I/O, and need not be connected to a terminal. This way we also protect the jobs from being stopped accidentally (by typing <CTRL-C> or <CTRL-S>). We can start, stop, and control them from LBL over a dial-up telephone line. Each detached job has its own log file for any system error messages. (One detached job logs all acquisition errors to disk.) Since these jobs run separately, they can each occupy 64KB of virtual space, and together can contain much more code than any single job. We also found it much easier to write multiple jobs than to struggle with one big job with overlays.

We use the TSX+ message facility to serialize the requests for GPIB access to the SERVER job and to return the requested data. This scheme may not be acceptable in situations where the requesting jobs have different priorities. (One way around this problem is to open a separate channel to the SERVER job, which it must check before checking the normal priority channel.)

We store the data in shared common blocks. This not only allows the data acquisition jobs to exchange information, but also allows us to access the data remotely to control and monitor the execution of the data acquisition software.

We use an IND control file as a command interpreter to execute the interactive and detached jobs. This allows us to monitor and analyze the data, and modify many parameters without interfering with the running MoWiTT data acquisition programs.

Some specifics of the software design:

The WATMTR and SERVER (when requested by FLOMTR) jobs read the ADAC pulse counters, which measure the flow rate in the cooling system and the wattage of heaters and fans in the test chambers. The ADAC pulse counters are 16-bit unsigned integers. In order to detect the counter overflow (which is actually the wrap around from 32767 to -32768 and from -1 to 0), we read the counts into I*4 words with the high order 16-bits set to zero. If we read the counts into I*2 words, F77's signed integers make it more difficult to correctly calculate the actual number of counts since the last reading. We never reset the counters because the few counts that could be lost between each read and reset operation would introduce cumulative errors.

We use the TSX+ shared run-time systems facility for common data buffers that multiple tasks can access simultaneously. When we generate the TSX+ system, we declare two run time system files in the form of .SAV (COMBLK.SAV and COMBUF.SAV files) which will never swap out of memory even if there are no jobs actively using them. We then create two common blocks, each exactly 4 KWords long, and include them in all the jobs that access these shared areas. We use the LINK/BOUNDARY (/Y) option to force the boundaries of the psects corresponding to the two common blocks to integral multiples of 4 KWords.

Each program uses the IADDR function in the RT-11 system library to get the two common blocks' starting virtual memory addresses, uses the IASRNT function from TSXLIB to associate the shared run time systems with the two common blocks, and then calls the MAPRNT function in TSXLIB to map the shared common blocks using two Page Address Registers (PARs). (Each program calculates which PARs to use from the high-order three bits of the virtual addresses of the first word of each common block.) The drawback to this technique is that we create a hole in virtual space that can be almost 4 KWords long. It may be possible to restructure the .psect ordering to fill most of the hole with some of the F77 OTS code, but we haven't found it necessary.

When we create MoWiTT data files, we use the function IDCLSF from TSXLIB to declare them as shared files, and give write access only to the jobs that write to the data files, and read access to other jobs that monitor the data.

We lock the data acquisition jobs into low memory using the TSXLIB function LKLOMY. They remain locked and never swap out of memory until they terminate. This prevents low-memory fragmentation.

While accessing the shared common blocks, each data acquisition task uses the TSXLIB routines TKCTL and RLCTL to take exclusive control of the CPU for very short periods. This prevents software race conditions between tasks which are reading and writing the same data.

Modifying the TSX+ system parameters:

We found it necessary to make the following changes to the TSX+ system parameters:

CACHE	set to 100 to provide faster operation of the floppy disks.
MAXCSH	set to 30 so we can cache all the logical and physical devices referenced by the data acquisition programs.
NMFCSH	set to 60 so we can cache all the files referenced by the data acquisition programs.
MIOBSZ	set to 2 to simulate 22-bit addressing for the 18-bit floppy controller.
MAXMC	set to 12 to allow the opening of message channels for all of our data acquisition jobs.
MSCHRS	set to 400 to match the length of the messages containing the data.
MAXMSG	set to 12 so the message queue can hold enough messages.
MAXMRB	set to 12 so the message-request queue can hold more requests for messages.
NCSILO	set to 150 so a job using CL-line I/O would not lose input characters while printing lines longer than the print output buffer.*
NCXOFF	set to 12 so remote jobs logged in from LBL would not lose characters when the user or the terminal typed <CTRL-S>.**

Notes:
*We had disabled spooling because it caused TSX+ stack overflows in an early version of TSX+.
**Related to using DZV11 interfaces for proper KERMIT handshaking. See next section.

PROBLEMS AND SOLUTIONS

We encountered a number of problems related to TSX+, RT-11, FORTRAN-77, the GPIB software, and ADAC's ADLIB library. We fixed some of them and found work-arounds for others.

TSX+ Problems (Sometimes it's really NOT your fault!):

We stumbled over a number of TSX+ system problems which caused system crashes:

(1) When we ran our software under TSX+ version 5.1, we intermittently got an SOF-Stack overflow error. (This happened whenever we tried to print with detached jobs running on a spooled system.) S&H fixed this problem in version 5.1C of TSX+ by increasing the size of the kernel stack.

(2) After we started our detached jobs, some of them DISAPPEARED! When we tried to duplicate the problem for S&H, it was IND that occasionally failed instead, while reading the command file to start up one of the detached jobs. (We were mystified by the error message: ?IND-F-No file accessed on channel.) S&H traced this to a bug in the TSX+ .CSTAT EMT implementation which returned extraneous bits in the high byte of the fifth word of the channel

status block (device unit number). (Internally, TSX+ had MOV'd a byte onto the stack without first clearing the high-order byte.) S&H fixed the problem in a field-test version of 6.2, but they still could not duplicate the original failure.

We sent an image copy of our data acquisition system disk to them, but they squeezed the disk before testing (so they could put some debugging files on it), and the vanishing-detached-job problem still did not show up. We sent them another image copy which they ran unsqueezed, and it DID fail. The problem was VERY obscure: a combination of the speed of our LSI-11/23 processor and the fragmentation level of the RL02 disk slowed down the start-up of the detached jobs enough for an internal race condition in TSX+ to cause the failure. (With an LSI-11/73 and a squeezed disk, S&H couldn't duplicate the problem.) S&H corrected both problems in version 6.2.

(3) We originally had a spooled TSX+ system. One of the interactive jobs we were running used a DLV11-J board through a CL line to drive the "SUNTRACKER" hardware to track the sun. We were receiving characters from this device through the CL line, and also occasionally writing to the serial line printer. We used REWIND 6 to flush the printer buffer. We then sometimes lost characters we were expecting to receive from the SUNTRACKER. (This caused unrecoverable errors because of the SUNTRACKER hardware's poor error-handling—it required manual intervention to restart the device.)

We couldn't understand TSX+ losing input characters on a 1200-baud DLV11-J line! S&H found the bug in DEC's LS handler (both LS.TSX and LS.SYS). The handler raised its priority to 7 in its interrupt-service routine, but didn't execute a .FORK. Then it entered a loop that picked up each character from the user-buffer, and discarded any NULs before outputting the next real character and returning from the interrupt.

Here's the combination of ingredients doctors recommend least: since the LS handler uses the $GETBYT system service under TSX+ (and RT-11 XM), handling each NUL on a slow (LSI-11/23) CPU can take a few hundred microseconds. Also, a Fortran WRITE to the printer followed by a REWIND 6 always outputs multiples of 512 characters (one logical block), even if only a small part of it is data; the rest is padded with NULs.

DLV11-J lines have only single-character input-buffering, and TSX+ could not service input character-interrupts on the SUNTRACKER's line while the LS handler's output interrupt-service routine was running at priority 7 picking up and discarding the several hundred NULs padding each logical block. By the time the CPU could service interrupts again, several character-times at 1200 baud had passed into history!

S&H solved this problem by patching LS.TSX (even though the problem originated in DEC's LS.SYS). A work-around for this problem would be to use RT-11 IWRITE (instead of Fortran WRITE plus REWIND 6) to send the exact number of characters to the handler without any extraneous NULs.

(4) We used KERMIT to transfer data from Reno to LBL. When we tried a 2400 baud modem, we noticed that the speed of the transfer was VERY slow (even slower than 1200 baud). Because of all the jobs running in Reno (some of which lock the CPU), the Reno KERMIT lost positive acknowledgements from LBL regularly. Then one of the two KERMITs would time out and retransmit its last packet. The effective throughput at 2400 baud including all the dead time waiting for time outs was less than the throughput at 1200 baud. We fixed this problem by replacing KERMIT's DLV11-J line with a DZV11 line, which has a 64-character hardware SILO buffer.

This caused another problem! We were losing characters when we typed <Ctrl/S> at LBL. (We use KERMIT as a terminal emulator to display output from programs running in Reno.) At 2400 baud, the LBL LSI-11 took too long to send back XOFFs when the CL input ring buffer got almost full, and/or the Reno LSI-11 took too long to stop sending characters after it received XOFFs. The combined delays exceeded the time required (four character times) to overflow the LBL CL input ring buffer. It isn't clear whether TSX+'s DZV11 output is too slow (at LBL), or TSX+'s burst-rate DZV11 input is less than 2400 baud (allowing the LBL input SILO to fill up before the Reno output stops), or if the detached jobs locking the CPU delay XOFF handling in Reno. We fixed this problem by increasing the value of the NCXOFF parameter in TSGEN from four to twelve, so the LBL LSI-11 would send XOFFs much sooner.

GPIB Problems (Beware IB.MAC's obscure comments!):

There was a problem with the National Instruments GPIB handler which crashed the entire system with a KTP-Kernel mode trap within a few seconds after we tried to access the GPIB. The IB driver time-out support code violated a basic rule of RT-11, which requires that the stack pointer not change between the .INTEN and .FORK EMTs. After the .INTEN, the IB handler interrupt service routine called an internal subroutine containing the .FORK required before issuing a .CTIMIO to cancel any pending device timeouts. The subroutine return address changed the stack pointer, so the .FORK returned into NEVER-NEVER LAND!

The way we fixed the problem was actually a work-around. We commented out all the time-out code. The acceptable consequence is that the IB driver will just hang the job if we ever have a problem with any GPIB device, like loss of AC power or a broken cable. Since we didn't need the timeout facility, we didn't take the time to fix it properly; we didn't feel it was our responsibility, and tried to get National Instruments to fix their own code, but they never have!

We also patched the IB handler's location 176 to zero with SIPP so we can always load the IB handler without causing a system crash, even if the hardware isn't installed. This is helpful during preliminary debugging on a development system without all the required hardware.

F77 Problems (An example of GREAT support!):

There was a bug in the FORTRAN-callable SECNDS function, which returns the time in seconds (REAL*4) between two consecutive calls. If we made two calls too close together, it returned the difference in time plus 24 hours. Multiware Inc. provided a patch which fixed this bug within days of our request.

ADAC ADLIB modifications (Our own trick to save 4 KW!):

We also modified the subroutines DEVICE and DAOUTP in ADAC's ADLIB library. As distributed, these subroutines use the RT-11 EMTs from SYSMAC, .POKE & .PEEK, which require the job to actually map the I/O page in order to access the hardware. This would have lost 4 KWords of address space by requiring jobs to map the I/O page.

Instead we modified the ADLIB routines DAOUTP and DEVICE to use the TSX+ EMTs (.PEEKIO and .POKEIO) which access the I/O page even if it is not mapped. We also wrote four routines called IOPOK, IOPOKB, IOPEK, and IOPEKB, using the TSX+ EMTs .PEEKIO, POKEIO, .PEKBIO, and .POKBIO, but haven't needed them yet. (They are the equivalent of the SYSLIB routines IPEEK and IPOKE but don't require mapping the I/O page.)

SUMMARY

We couldn't have done it without TSX+, or without help from the many people (see acknowledgements below) who struggled with us throughout our most obscure difficulties.

TSX+ eventually proved to be a very reliable real-time platform for our complex data acquisition, analysis, control and remote monitoring project.

ACKNOWLEDGEMENTS

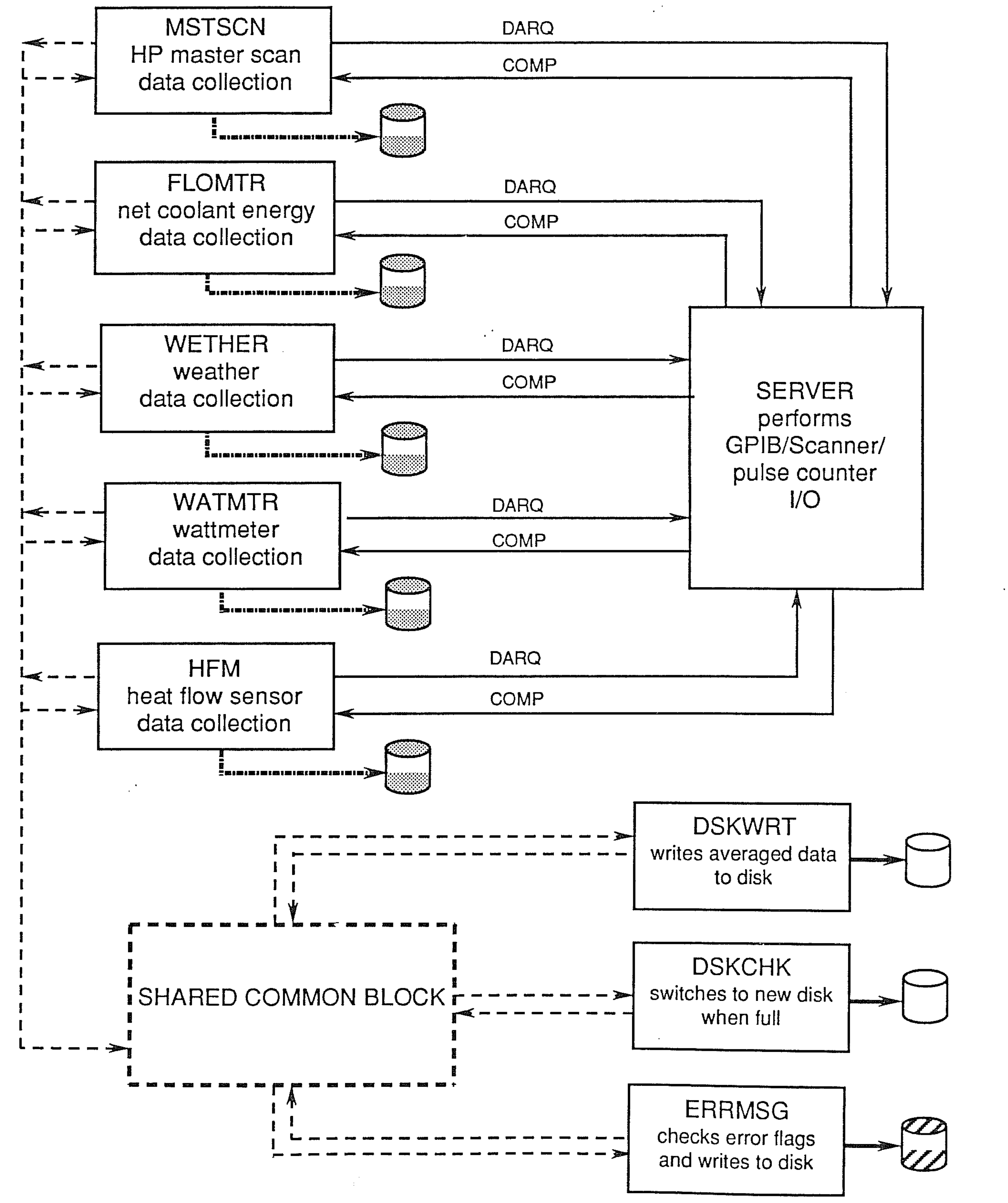
We would like to acknowledge the following people for their assistance and cooperation:

Jan Bramlet, formerly of S&H Computer Systems, Inc.
Jim Crapuchettes of Omnex Corporation.
Bob Walraven of Multiware Inc.

and the present and former members of the MoWiTT technical staff, Dennis DiBartolomeo, Mary Hinman, Joe Klems, Steve Lambert, and Michael Streczyn.

This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Building Technologies, Building Systems and Materials Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

DETACHED JOBS FOR DATA ACQUISITION
RUN CONTINUOUSLY

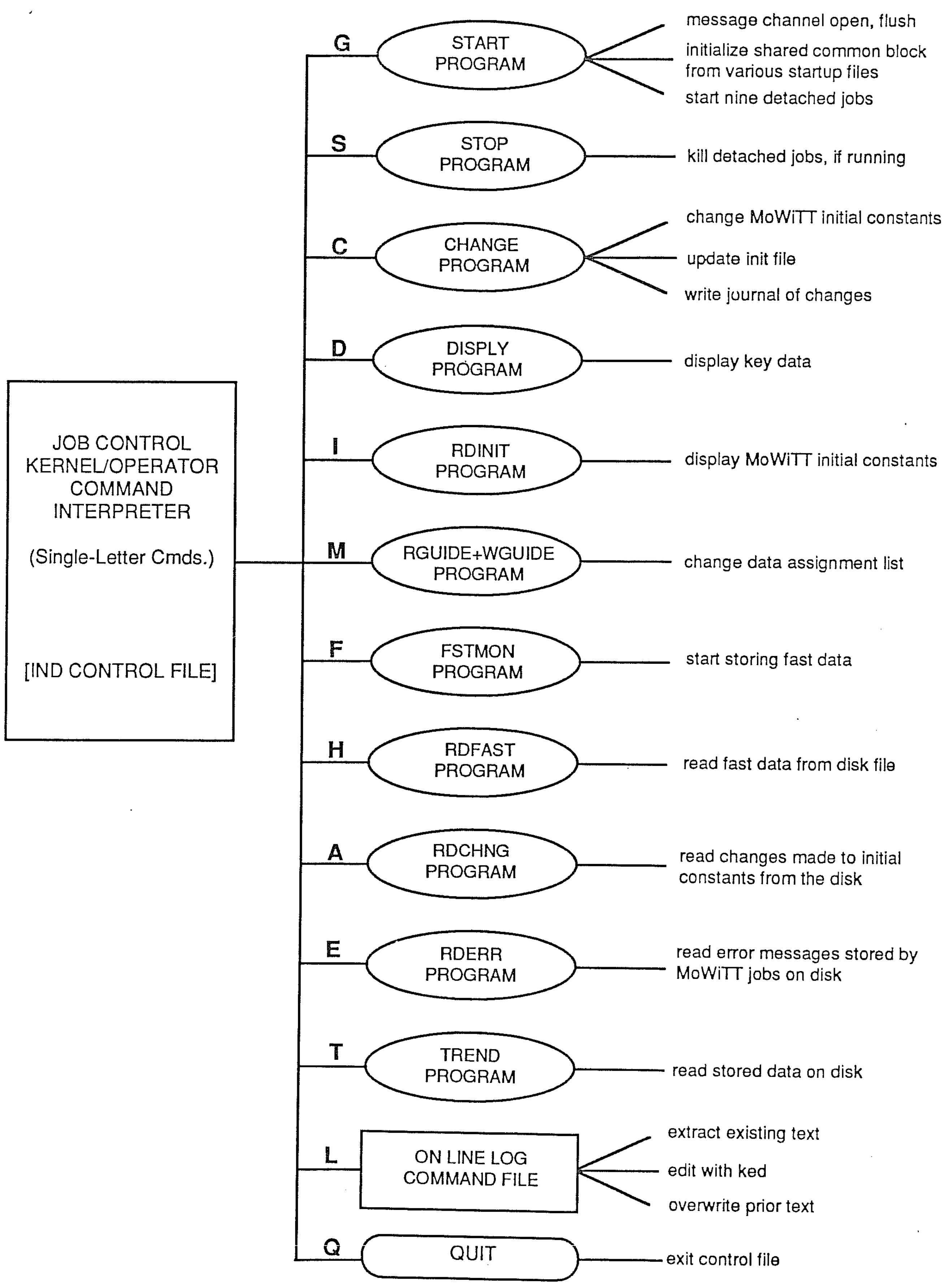


DARQ = Request action after a given delay

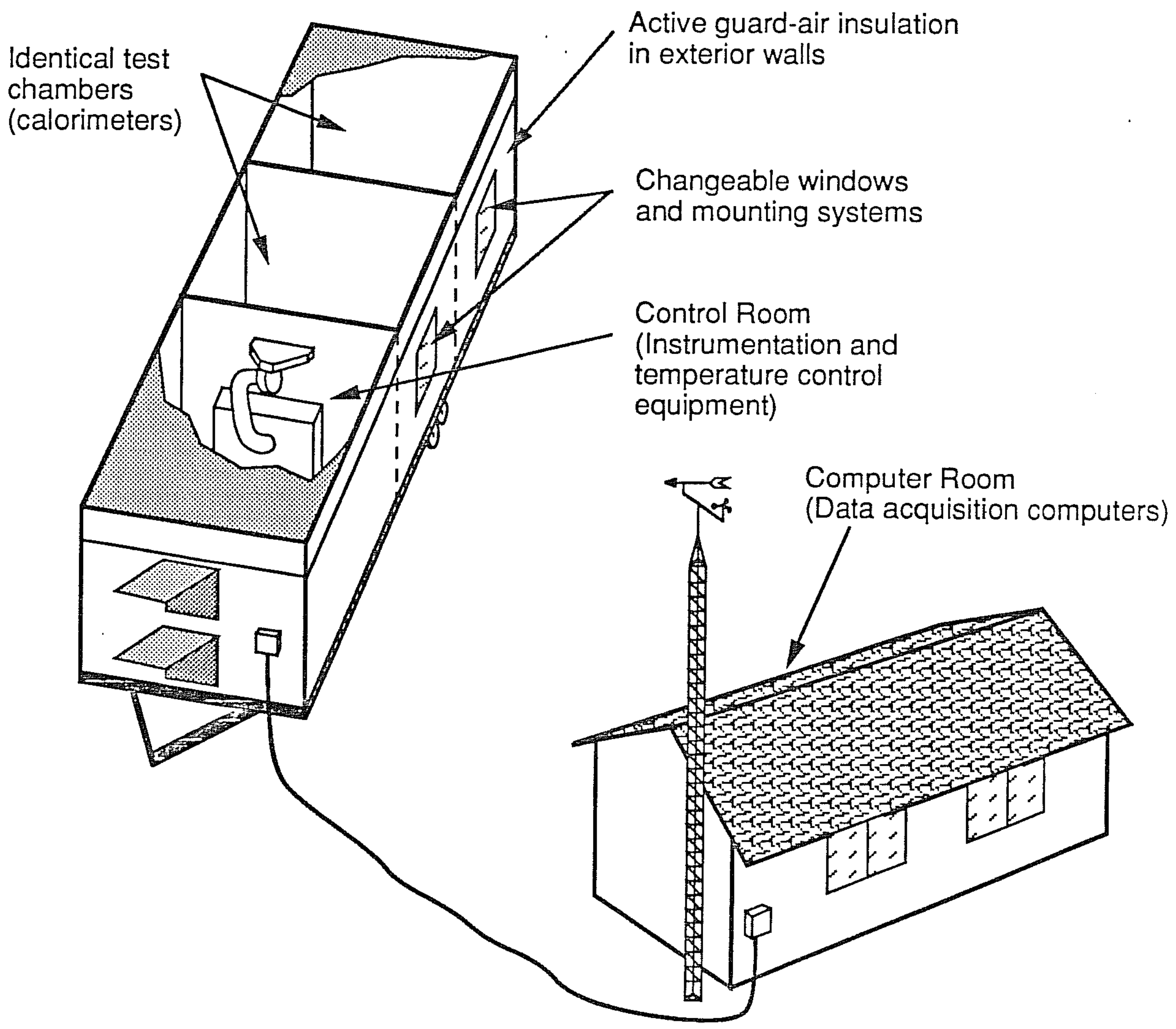
COMP = Requested action completed;
(contains requested data)

———	data passed through message channel
– – –	data passed through shared common block
-·-·-·-·	optional fast data recording
▬▬▬	normal data recording

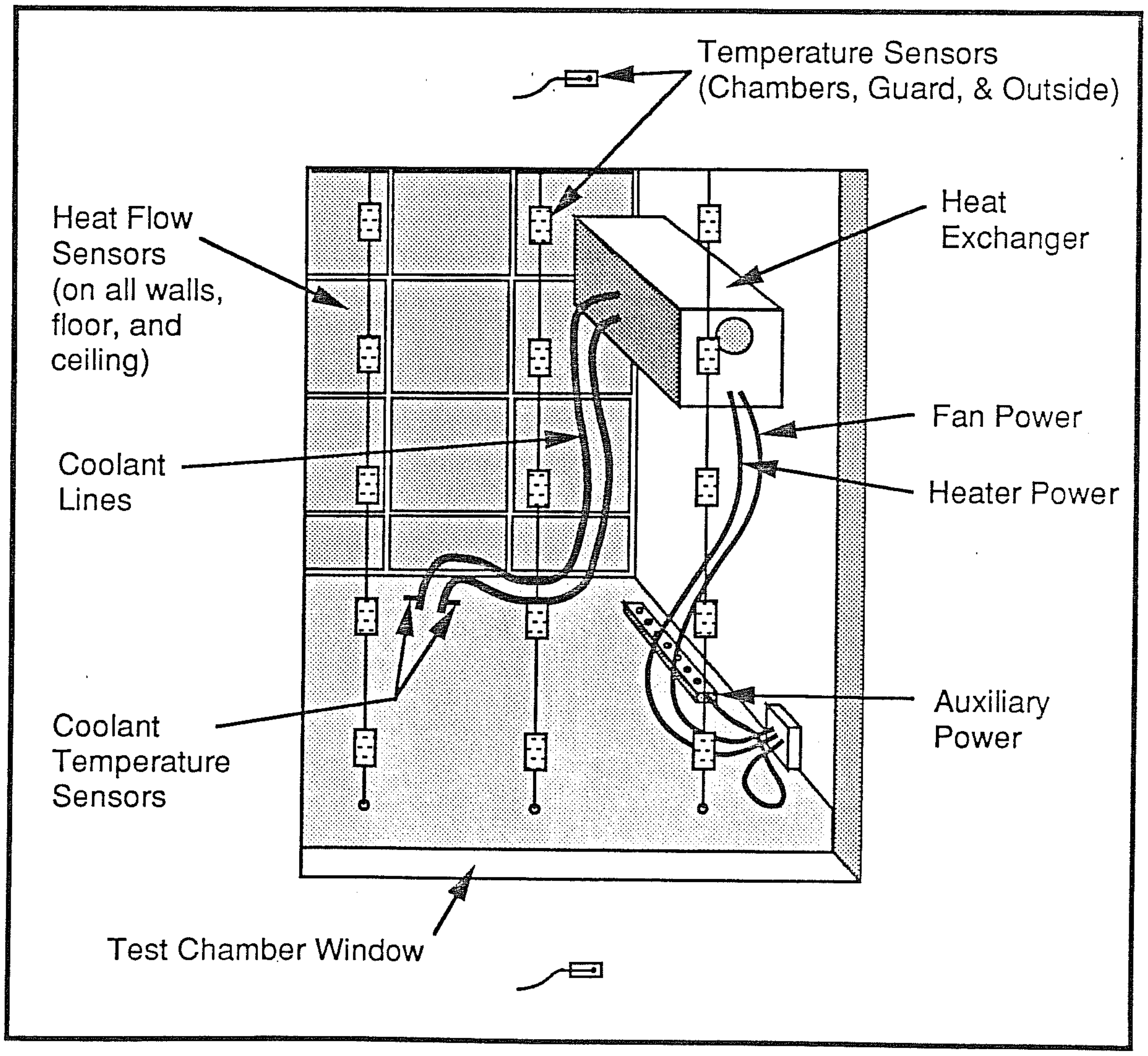
INTERACTIVE CONTROL JOB (CNTROL.COM)



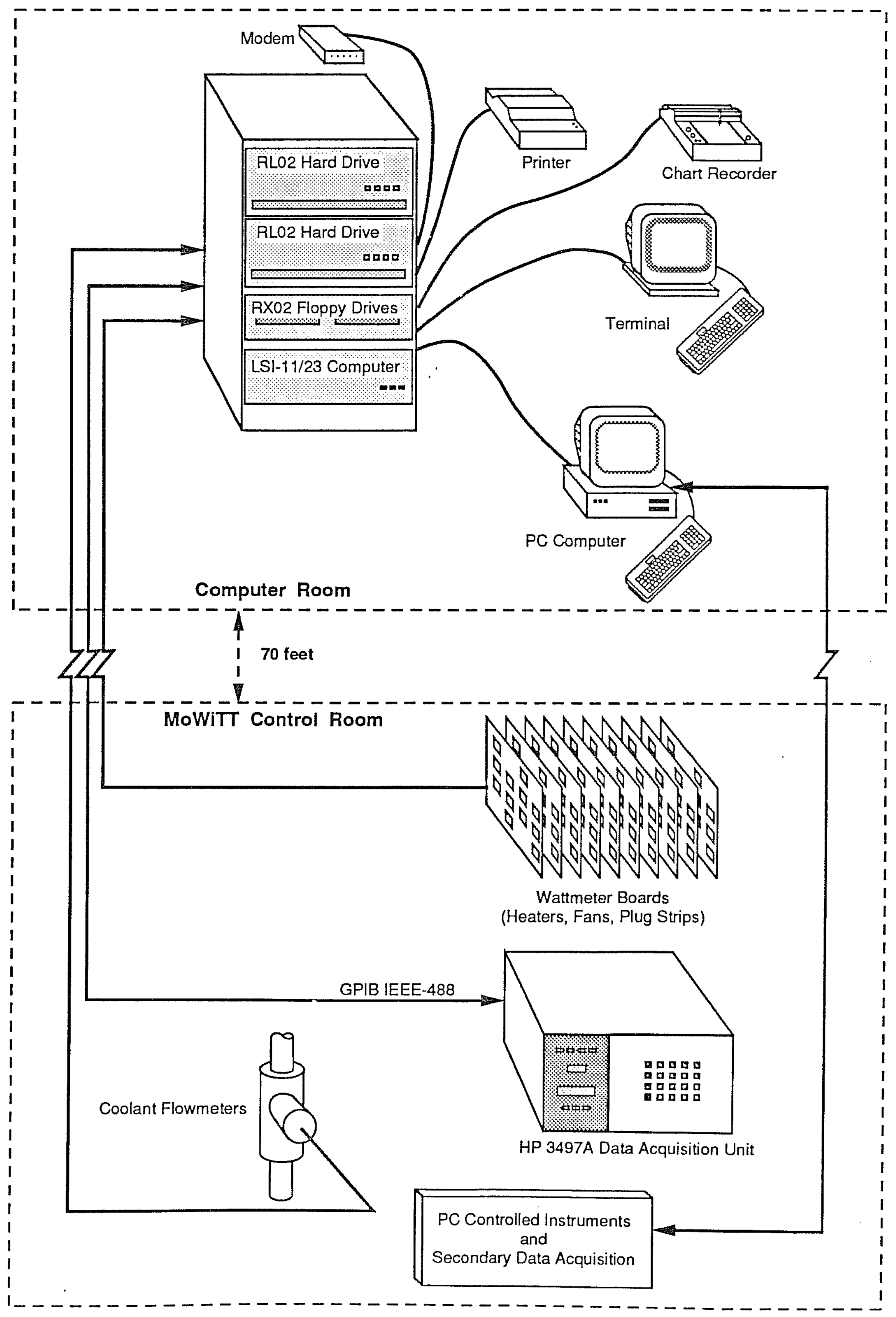
Mobile Window Thermal Test Facility (MoWiTT)


Identical test chambers (calorimeters)
Active guard-air insulation in exterior walls
Changeable windows and mounting systems
Control Room (Instrumentation and temperature control equipment)
Computer Room (Data acquisition computers)

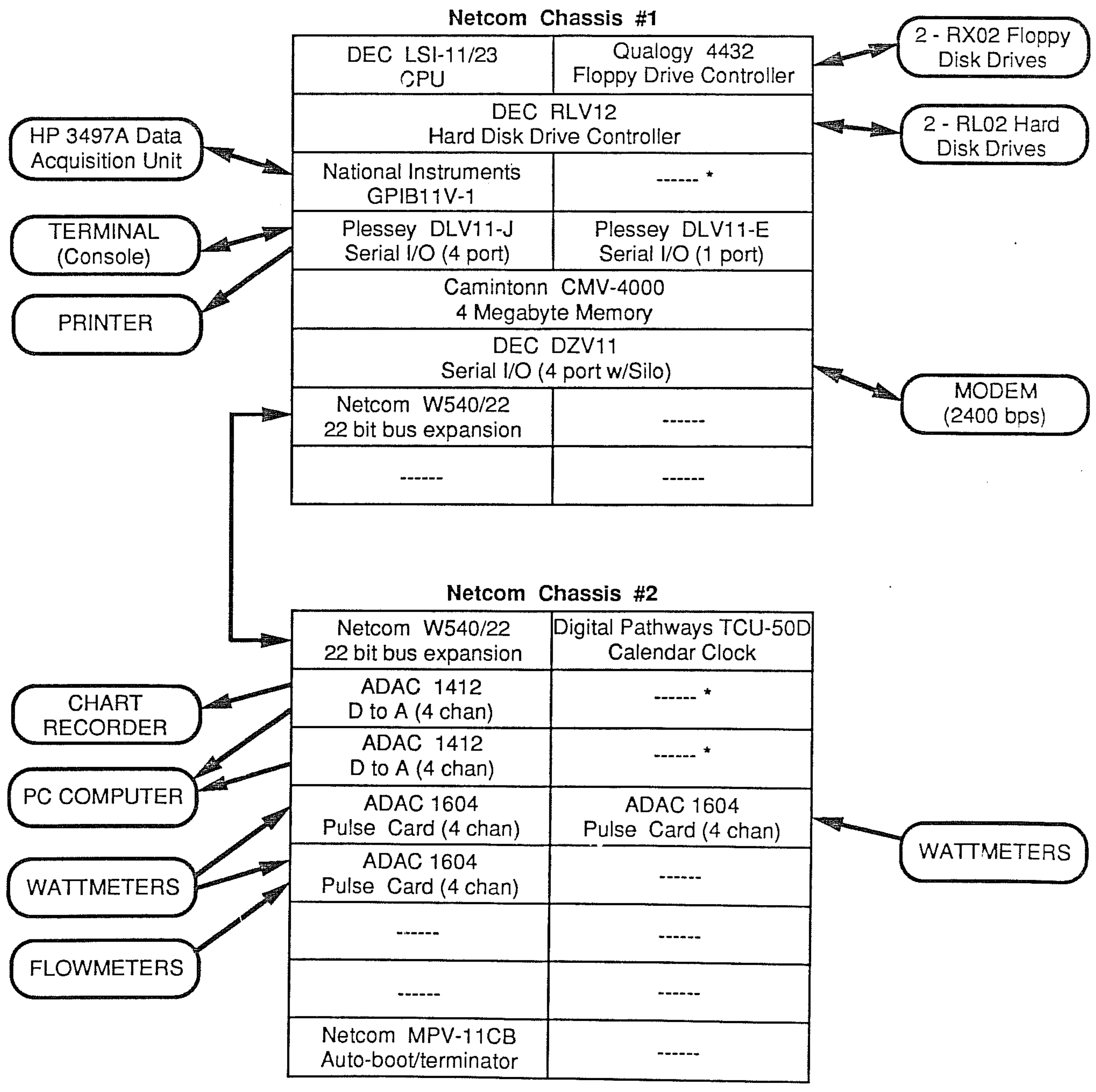
MoWiTT Test Chamber


Temperature Sensors (Chambers, Guard, & Outside)
Heat Flow Sensors (on all walls, floor, and ceiling)
Heat Exchanger
Coolant Lines
Fan Power
Heater Power
Coolant Temperature Sensors
Auxiliary Power
Test Chamber Window

MoWiTT Data Acquisition System



MoWiTT LSI-11 Computer Backplane


Netcom Chassis #1
DEC LSI-11/23 CPU
Qualogy 4432 Floppy Drive Controller
DEC RLV12 Hard Disk Drive Controller
National Instruments GPIB11V-1
------ *
Plessey DLV11-J Serial I/O (4 port)
Plessey DLV11-E Serial I/O (1 port)
Camintonn CMV-4000 4 Megabyte Memory
DEC DZV11 Serial I/O (4 port w/Silo)
Netcom W540/22 22 bit bus expansion

------
2 - RX02 Floppy Disk Drives
2 - RL02 Hard Disk Drives
HP 3497A Data Acquisition Unit
TERMINAL (Console)
PRINTER
MODEM (2400 bps)
Netcom Chassis #2
Netcom W540/22 22 bit bus expansion
Digital Pathways TCU-50D Calendar Clock
ADAC 1412 D to A (4 chan)
------ *
ADAC 1412 D to A (4 chan)
------ *
ADAC 1604 Pulse Card (4 chan)
ADAC 1604 Pulse Card (4 chan)
ADAC 1604 Pulse Card (4 chan)

Netcom MPV-11CB Auto-boot/terminator

CHART RECORDER
PC COMPUTER
WATTMETERS
FLOWMETERS
WATTMETERS

* Unused (C-D interconnect)